Date: 11-18-03 Express Mail Label No. EV 214950315 US

Inventors: Nicholas Stamos, Seth N. Birnbaum, Tomas Revesz, Jr., Donato Buccella, Keith A. MacDonald, Dwayne A. Carson and William E. Fletcher

Attorney's Docket No.: 3602.1000-003

DIGITAL ASSET USAGE ACCOUNTABILITY VIA EVENT JOURNALING

RELATED APPLICATION

This application is a continuation of U.S. Application No. 10/655,573 filed on September 4, 2003, which claims the benefit of U.S. Provisional Application No. 60/442,464 entitled "Method and System for Adaptive Identification and Protection of Proprietary Electronic Information," filed on January 23, 2003. The entire teachings of the above-referenced application are hereby incorporated by reference.

BACKGROUND OF THE INVENTION

Data security has been a significant issue facing system administrators since almost the inception of the data processing industry. Most computer users recognize the possibility of theft or misuse of data by unauthorized outsiders. The terms "hackers" or "crackers" are often used to describe such outsiders who attempt to gain access to a system, and who are typically not involved in any way with an organization's operations, its internal employees or systems. Many different solutions already exist to protect an organization's data processing infrastructure from this kind of threat. These include physical access control, firewalls, sniffers and other network monitors, data encryption, intrusion detection systems and other solutions. These solutions are generally recognized as being adequate for their intended purpose most of the time.

However, there is a second class of computer users that also pose a security threat. Protection from these unauthorized insiders requires a different approach, but one that is also well known. Almost since the inception of disk-based storage systems, the concept of access control has been applied to limit the ability of certain users to access certain important files. Using these techniques, now a universal feature of in any Operating System (OS), a desktop and/or network file server can provide for limited read, write, public, private and other types of access to files, directory structures and the like, depending upon permissions granted to particular users. Permissions can be attached to user accounts by a system administrator, based on their need to know, departments in the organization of which a user is a member, and so forth.

Even when users obtain access to only a portion of a system, however, they can still use a variety of techniques to steal and/or damage information. These can include simple browsing for unsecured information in a network, and/or removal or deletion of information made available as a result of poor security practices. More sophisticated rogue users will employ network packet sniffers and/or spying software. Fortunately, a variety of approaches, such as centralized document and digital rights management systems, network auditing, and file management tools, are effective tools against unauthorized use by insiders.

For example, U.S. Patent 6,510,513 issued to Danieli and assigned to Microsoft Corporation describes a security and policy enforcement system that utilizes a series of transactions between a server and a client using electronic security certificates. A first client generates a request for access to data by submitting a security certificate containing a digest to a trusted arbitrator server. The trusted arbitrator authenticates the first client's credentials and returns the security certificate. The data and security certificate are then combined to create a distribution, which, in turn, is acquired by a second client. The second client extracts the security certificate and generates a digest from the data in the distribution. If the digest from the second client matches the digest from the first client, then data is considered to be valid. Depending upon the certificate type and a policy level, the trusted arbitrator server can provide services such as notification of improper usage.

U.S. Patent 6,427,140 assigned to Intertrust Technologies is another type of digital rights management system. A system such as this is intended, for the most part, to protect the rights of various participants in a transferring sensitive data, such as in an electronic commerce or other electronic facilitated transactions.

SUMMARY OF THE INVENTION

Neither of these solutions do much to protect misuse of information by authorized insiders. This class of users has a trusted status, as they are supposed to have access to important data files to carry out their assigned tasks. Thus, they are routinely granted permission to use such information on a daily basis, and their use is not normally suspect. The problem comes when a class of trusted users abuse that trust by copying and/or distributing sensitive information to outsiders or other unauthorized people. Such events can happen quite easily and with increasing frequency when a disgruntled or departing employee wishes to damage an organization.

What prior art security systems fails to account for is the fact that once granted access to sensitive information, it is quite easy for authorized users to distribute it in many different ways. The proliferation of Internet connections, e-mail, instant messaging, removable media storage devices, such as Compact Disk-Read Write (CD-RW) drives, Universal Serial Bus (USB) type memory and storage devices, and the like, it makes it a trivial task to copy vast amounts of information almost instantaneously. Other peripheral devices, such as wireless modems, wireless local network cards, portable computers, Personal Digital Assistants (PDAs), network tunnels, and the like, provide further vehicles by which an authorized user may distribute copies of files outside of the trusted system environment. Even an act of printing the contents of a file is a potentially damaging event.

This is the case even when sophisticated file management and access control systems are employed to control access to and even monitor usage of files. The root of the problem stems from the fact that once an authorized user opens a file, its contents

are no longer controllable. Specifically, copies of the file contents may be taken "out of" the controlled environment of a network or file management system.

The present invention is intended to address security problems that originate with authorized users abusing their authority, by providing a usage accountability model for data security.

In particular, an autonomous, independent agent process, such as running in the background of a client Operating System (OS) kernel, interrupts requests for access to resources. Such resource access requests may include, for example, requests to read a file, open a network connection, mount a removable media device, and the like). Since access is detected at the OS kernel level, tracking of resource utilization will occur regardless of whether the original access request originated from an application program that is being executed by an end user, indirectly by applications on behalf of users, or even by system requests made independently of application software.

The autonomous independent agent process contains sensors that capture low level system events. These may include, for example, operations such as file read, file write, file copy, clipboard cut, clipboard copy, CD-RW access, TCP/IP network message inbound, TCP/IP network message outbound and the like.

Low level events are then associated with one or more file names (handles) and filtered against an approved list. Thus, the raw events are filtered to remove references to files such as operating system files (.EXE, .DLL, etc.) and the like that do not contain sensitive application data. Only events relating to application files that may contain sensitive data are thus further tracked.

The filtered results are then bundled together and sent securely to a journaling server. The journaling server unbundles the list of events and stores them in an event database. The journaling server also periodically looks at a series of events in order to recognize an aggregate event as a possible abuse of trust situation. Such aggregate events are also then typically also added to the database.

For example, an aggregate "FileEdit" event might be reported by the journaling server when a user has opened and modified a sensitive financial document, with that user then printing the document before renaming it and saving a it to a newly attached

USB hard drive. A set of reports can then be generated from journaled aggregate events to provide a comprehensive understanding of how files were accessed, used and communicated by individual users in an enterprise. Summary and trend reporting, for example, can show the volume and type of information that flows, and possible links between aggregate events for particular suspect users based on a variety of criteria.

Activity journals can also be sorted by user, a file, application, network connection, storage media, and the like. The result is an audit trail that can be used for a variety of purposes to determine, for example, which files have been attached to emails sent through a personal email server, which users have access specific client files and which documents have a recently departed employee burned to a CD-RW or printed to a home printer in the last month, or other possible abuses of authority.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Fig. 1 is a diagram illustrating traditional security perimeters in a data processing system and a point of use perimeter that can be implemented with the present invention.

Fig. 2 is a diagram illustrating how events at client computers and file servers in a network are sensed, bundled, and sent to an activity journal server.

Fig. 3 is a process flow diagram illustrating the invention more particularly.

Fig. 4 is a table of possible low level atomic events.

Figs. 5A- 5C are a table of higher level aggregate events.

Figs. 6A-6C show reports that can be generated by the invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

Fig. 1 is a diagram of a typical computer network 100 which consists of client devices 102 and servers 104 connected via local area network and/or inter-networking equipment. Connections to an outside network, such as the Internet 108, are made through devices such as routers or gateways106. Connections through the Internet 108 can be also made to external computers 110 that form a trusted extranet.

A traditional security model is used to prevent access by an untrusted outsider 110 to devices 102 and/or file servers 104 within the protected network 100. A network perimeter 120 is thus associated with network points of access, such as through router 106 and specifically at a firewall 107. The firewall 107 can thus prevent attempts by unauthorized users of outside computers 110 to access information stored in the server 104 or otherwise manipulate the local computers 102. Firewalls 107 can also establish a perimeter 120 for outgoing access such as, for example, by users attempting to access certain undesirable outside computers 110 that contain restricted or harmful websites, game servers, and the like.

Rather than establishing a perimeter at external points of physical access to a network, the present invention establishes a perimeter of accountability for file usage. The accountability model can not only track authorized users of the computer 102 accessing files stored on a local server 104, but more importantly also monitors passage of such files to peripherals that distribute or record information, or other possible abuse events.

Such possible abuse events may occur whenever a user accesses devices which are not visible to or controllable by a local file server 104 or firewall 107. These events may include writing files to uncontrolled media such as CD-RWs 204, PDAs 206, USB storage devices 208, wireless devices 212, digital video recorders 214, or even printing of files. Other suspect events can include running external Peer to Peer (P2P) applications 201, sending files via external e-mail applications 202, uploading files to web sites via the Internet 108, and the like. Thus, the invention can provide an enterprise-wide journal of all file, application and network use. As will be understood

shortly, the heart of this journaling approach consists of a high level contextual stream that characterizes user activity as it occurs at the point of use, such as the desktop 102 or file server 104.

Turning attention to Fig. 2, the activity journalling process will now be described in more detail. An agent process 300 is interposed between an Operating System (OS) 301 and applications 308 as they run on clients 102 and/or servers 104 within the network 101. The agent process 300 is used to detect and track file, printing, clipboard, and I/O device operations, such as file read or write operations, or network data transfers.

While the clients normally include desktops 102-1 which have a direct wired (or wireless) connection 109 to the local network 101, the agent 300 may also run on disconnected client computers such as laptops 102-2, making a report of events once a connection is eventually made to the network 100.

In a manner that will be described shortly, the agent 300 reports atomic events 350 to an activity journaling process typically running on an activity journaling server 104-2. The journaling server 104-2 processes atomic event data and coalesces it into what are called aggregate events 360. Aggregate events 360 are detected when a certain predetermined sequence of atomic events occurs. Each aggregate event 360 is thus composed of one or more atomic events 350 that conform to some predetermined pattern indicative of activity that should be monitored.

Specific types and/or sequences of atomic events 350 that lead to aggregate events 360 will be described in detail later. It should be appreciated here, however, that the particular events reported and their aggregation types depend upon the specific activities sought to be monitored.

To protect the network completely, typically the agent process 300 would reside on all desktops 102 and file servers 104 associated with an enterprise's networks. The activity journaling server 104 and agent process 300 may communicate through secure, networking based applications such as the Microsoft ".NET" infrastructure or other secure networking systems. A management console 102-5 permits access to the database stored in the journaling server 104-2, and is used specifically to provide risk

compliance, forensic reporting, and similar reports 310 to administrative users of the system.

Fig. 3 is a more detailed view of the client agent 300 and journaling server 104-2. These elements particularly consist of one or more sensors 500, file filter 520, event coalescing aggregation 530, network connection 550, database 560, and high level event aggregation 570 to perform an event detection and aggregation. It should be further noted that the agent process 300 can also provide real time evaluation and potentially enforcement of rules.

The journaling server 104-2 may typically run within a Windows 2000 Server environment having a secure .NET framework. The journaling server 104-2 also has access to a database, such as Microsoft SQL Server 2000 for example, to provide record storage and retrieval functions. It is to be understood, of course, that the processes described herein can be implemented on other types of operating systems, server platforms, database systems, and secure networking environments.

As already mentioned, the agent 300 typically runs as a kernel process in a client Operating System (OS). For example, the agent 300 may run within the kernel of Microsoft Windows 2000 or Windows XP. Autonomous operation of the agent 300 provides for detection of atomic events 350 even when client 102 is disconnected from the network 100. Any such events are reported when the client 102 is reconnected and can communicate with the journaling server 104-2.

In a preferred embodiment, the agent 300 will run multiple services under Windows so that if one service is stopped by a malicious user, the other one may restart the other process. The process is also hid from a task manager or similar processes in the operating system and will be able to work with safe mode boot features in order to guarantee full protection.

Turning attention to the agent 300, atomic event sensors 500 provide atomic events as output when action typically associated with Input/Output (I/O) drivers are intercepted at the OS kernel. The agent process 300 is therefore transparent to the end user and tamper resistant. The intercept may, for example, occur during an I/O Request Packet (IRP) in an interruptible kernel. The sensors 500 may include, for example, file

operation sensor 502, network operation sensor 504, print queue sensor 505, clipboard sensor 506, Application Programming Interface (API) spy sensor 508 and other sensors. Events may be provided for example, by Windows services and kernel level drivers.

Data collected with an event depends on the event type, but can include:

- For invoked applications, the identity of the invoking process, executable name, start time, end time, and process owner
- For user operations, such as log on or log off, the time and user identification (ID)
- For file operations, source / destination file name, operation type (open, write, delete, rename, move to recycle bin), device type, first and last access time
- For network operations, source / destination address, port and host names, start/end time stamp, bytes sent and received, inbound and outbound data transmission times
- For CD-RW operations, file names, start/end times and amount of data transferred
- For printing operations, full path or file name, event start time or print job name
- For clipboard operations, destination process ID, event start time, full path of filename involved
- For other high level operations, such as access to removable storage media, file name, device ID, time of day, bytes transferred, and the like

An approved file filter 520 operates to automatically filter the dozens of inconsequential events generated by standard calls to system files. For example, it is quite common for many different .EXE and .DLL operating system files to be opened and accessed repeatedly in a typical executing Windows application. In order to reduce

the data flow to the journaling server 104-2, the file filter 520 uses an approved file list 522 to filter atomic (raw) sensor events 510.

The approved file list 522 may be implemented by a list of file names associated with events. However, in a preferred embodiment, the well known MD5 algorithm is used to generate a hash code for each file name. The MD5 hash code for a filename associated with an event is then matched against the approved list 522, rather than the complete file handle, to speed up the filtering process. Thus, only events associated with unapproved files are passed down to the coalescing stage 530.

The next stage is an atomic event coalescing stage 530 that attempts to aggregate atomic events 510. The coalescing function further filters atomic events 510 associated with or related to a single user action between the agent 300 and the journaling server 104. In general, applications frequently read small chunks of a file and not the entire file at the same time. For example, a user may open a 2 MegaByte (MB) spreadsheet file. However the OS may at a given time actually only access chunks of the spreadsheet file that are much smaller than that, such as 5 or 10 KiloBytes (KB) at a time. Thus, a typical pattern of access is to see a file open atomic event, followed by multiple read atomic events to the same file. If this sequence of atomic events is seen from the same process and the same executable with the same thread ID and the same file handle, event coalescing 530 will thus count only a single "FileOpen" event. In a preferred embodiment, there is a time attribute associated with event coalescing 530 such that if a time limit typically measuring in minutes of time is exceeded, at least one event will be reported between raw level events.

The coalesced events are then grouped together in bundles 540-1, 540-2 . . ., 540-n. A bundle 540 consists of a number of events that are grouped together for the convenience of transmission from the client 300 to the server 104-2.

Communication between the agent 300 and journaling server 104-2 preferably takes place over a fault tolerant, encrypted, asynchronous communication channel 550, such as a Hyper Text Transfer Protocol Secure (HTTPS) channel. For example, the Public Key Infrastructure (RSA/PKI) available from RSA Security, Inc. can be used for symmetric encryption. The agent 300 holds a service certificate (server public key) that

it uses to encrypt one time session keys, on a per packet basis, to implement symmetric cryptography.

Compression and other data reduction techniques can also be applied to the bundles prior to their transmission over the network connection 550. With file filtering 522 and atomic event coalescing 530, it is expected that the size of the activity journal to be communicated to the server 104-2 typically is on the order of only about 150 Kb per user per day.

On arriving at the journaling server 104-2, bundles 540 are decompressed and decrypted, returned to their original state, and placed in the database 560 as the atomic event table. This table holds a de-multiplexed version of low level coalesced events so that they may be processed as a single stream.

A high level event aggregation process 570 then periodically reads events from the database table 560 as a stream and determines if high level aggregate events have occurred. This can be done by running queries on the database 560 to determine if a sequence of atomic events has occurred in patterns than are defined in advance.

A comprehensive list of typical high level event patterns is shown in Fig. 4. For example, 43 different action types, some of which are low level atomic events and others which are high level aggregate events, are defined in the preferred embodiment. A given event is composed of several fields in the database, including perhaps an action type 571, level 572, event category 573, event name 574, event table ID 575, action detail 576, action detail value 577, and discriminants 578.

Event categories are associated with each event type. For example, in an event category "file", event names include file read, file write, file rewrite, file copy, file rename, file delete, file move, file recycle, file restore. Similarly, network related events are TCP/IP inbound, TCP/IP outbound, USB inbound and so forth.

A scope is also associated with each event type. A scope is defined as either being a thread, process, login, machine, or all type scope. For example, "process" scope is an event that is consolidated into a high level event in the same process but not necessarily executing the same thread. "Machine" means that a reboot could occur between two events that occurred on the same machine.

Attributes commonly recorded for all high level events include an action type, an event count, bytes read count, bytes written count, event start, event end, and other possible actions. Source and destination hold numerous other attributes including the file, path, process, thread, and application identifying information that performed the event.

Other types of system events may include print events, CD events, clipboard, user and machine events. The final type of low level event may be process events including process start and process end.

The database 560 will eventually include a series of various events, such as file events, network events, print events, CD events, user events, machine event, process, machine device and other events.

High level aggregate events are created by detecting a combination of the occurrence of low level events. More particularly, a high level aggregate event (action types 26-42) is determined after seeing a specific sequence of lower level events (action types 1-25). For example, action type 26 is a high level event called "FileEdited". This is an aggregate event that determines when a file has been edited. As the table indicates, the high level event aggregated process 570 may detect that a particular process, thread, and file has performed one or more reads to a particular file handle, followed by a write operation to the same process, thread and file handle. The event is then defined as an aggregate "File Edited" event.

Aggregate events are defined in greater detail in Figs. 5A, 5B and 5C. For example, a "Clipboard to File" aggregate event 510 is defined as detecting a clipboard cut or copy followed by a clipboard paste to file operation.

Similarly, a "BurnFile" event is associated with detecting a CD write atomic event followed by a file read atomic event. Thus, if a series of file reads are detected from one file handle, followed by a series of CD write events with the same process, the application is recognized as having written a file to a CD-RW.

Numerous other aggregate events are possible; the list in Figs 5A, 5B and 5C is only meant to illustrate a few of the many possibilities.

Fig. 6A is an example summary report that can be generated from the aggregate event. In particular, statistics can be taken on a daily, weekly or other basis to list when digital assets have been removed to uncontrolled media, when digital assets have moved to external networks or to other uncontrolled environments. Reports can be provided in this summary form, or can be of course provided in more detailed format, as shown in Fig. 6B, arranged by department and even by a particular user. Patterns of unexpected behavior can then be determined from such reports.

Further detail such as arranged by a particular user can be provided in a report as shown in Fig. 6C. Here, a particular user, Albert Grimley, is seen to have made copies of design specification files, sales pitches, customer lists, product overviews, and marketing slides. If such activities are not normally expected to be authorized for Mr. Grimley, such as for example, if his job responsibilities are to assist the engineering development team, and he is not in the marketing department, activities such as copying customer lists, sales pitches and marketing slides might be considered to be suspect, requiring further action by the organization's management.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.